# Network Frame Monitor

*Local application*

Wed, Jun 4, 1997

The FMON local application is designed to aid network frame diagnostics. It can capture a part of network frames received or transmitted to/from a selected target node, or from all nodes. It also checks for duplicate frames by monitoring the *identification* field in the IP header.

### *Traditional embedded network diagnostic*

In each IRM or local station system, network frame diagnostics are included that write into the NETFRAME data stream (DSTRM entry #0) a short record of each network frame that is received or transmitted by that node. The record contains the node#, the size of the frame, a ptr to the frame buffer holding the frame, and the calendar time of the frame reception or transmission, with resolution down to half milliseconds since the start of the current 15 Hz cycle. A page application NETF allows viewing of the contents of this data stream. It can also allow viewing frame contents, but this part can only work if one asks to view a given frame before the circular buffer of frames is overwritten by successive receive or transmit frame activity. For busy systems, especially including IRMs that connect to ethernet, one may have only a few seconds before the buffers are overwritten. This makes it very difficult to do some kinds of frame diagnostics using this embedded tool, when it is necessary to view frame *contents* at leisure.

### *New method of capturing frames*

The FMON local application runs in a system for which frame diagnostics are needed for frames associated with communication of that node with another node. The approach in using FMON is that of doing an experiement, rather than operating a typical utility. These are the FMON parameters available on Page E:

```
E LOC APPL PARAMS 06/03/97 1438
NODE<0509>  NTRY<28>/64  H<0508>
NAME=FMON   CNTR=47  DT= 0.5  MS
TITL"NET FRAME MONITOR       "
SVAR=00044D9A     05/06/97 0931
ENABLE  B<00EF>*FMON ENABLE
NODE#    <E071>
ON      B<00EE>*FMON CAPTURE ON
MAX INDX <2000>
DIAGPHI  <0030>
DIAGPLO  <0000>
W OFFSET <0011>
FLAGS TR <0003>
```

First there is the usual enable Bit# needed for each local application instance. The node# parameter identifies the target node for which frame communications are of interest. In the case of IP-based communications, this is a pseudo node# that corresponds to an IP socket, the combination of an IP address and a UDP port#. The used for ethernet. The middle two digits identify a 16-byte entry 02–FF in the IPARP table, and the least digit is a port# index that refers to a UDP port# via the associated port# block that is allocated for use while an IPARP table entry is active. An entry times out after no activity is observed for about an hour. Port# index values of 1–F signify valid UDP ports in current use. An index value of 0 means a port# is not used, such as for the case of an ICMP ping request/reply message or an IP datagram fragment. So, if a pseudo node# is used as a target node# parameter, it really refers to a node and port combination. In order to find out what pseudo node# value to use here, one may have to check the traditional NETFRAME diagnostics during a trial run.

The ON capture Bit# parameter allows enabling or disabling the frame capture logic without disabling the local application itself. This is convenient because one needs to check the FMON static variables—allocated dynamically when the local application enable bit is set—to get information about where the frames are currently being captured. The maximum index parameter refers to the size of the memory available for capturing frame contents. A 64-byte data structure is used for frame capture; thus, only a portion of the frame may be captured. In this example, the maximum index parameter of $2000 means that there is room in memory to capture parts of 8K frames. The two following parameters together comprise the 32-bit base address of memory that is to be used for the frame capture data. For IRMs it is common to use $300000–3FFFFF for this purpose. This is the fourth megabyte of memory on the CPU board, currently not used for system purposes.

The word offset parameter allows capturing parts of frames not beginning with the start of the frame header. For example, for ethernet, an offset value of $07 will skip the 14-byte frame header. For IP frames, a value of $11 will skip both the frame header and the 20-byte IP header. A value of $15 would additionally skip the 8-byte UDP header, allowing capture of frame contents beginning with the start of the UDP datagram contents. the captured data is actually limited to 56 bytes, as the first 8 bytes of the 64-byte record is for the calendar time of the frame reception or transmission. The "TR FLAG" parameter refers to whether receive, transmit, or both receive and transmit frames are to be captured. The least significant two bits are used for this purpose:

| | |
|---|---|
| 0 | none |
| 1 | receive |
| 2 | transmit |
| 3 | both receive and transmit |

Note that the use of a 64-byte structure to hold captured frame contents is a convenient match to the Memory Dump page application—not an accident. One can advance through successive frames using that diagnostic page application and very quickly identify changed fields of interest.

The FMON program builds on the traditional frame NETFRAME diagnostics in order to provide its functions. It uses a local data request to monitor the records written into that data stream each 15Hz cycle. When it gets a match for capture, it uses the buffer ptr to grab the frame contents that are to be captured, during that same 15 Hz cycle, thus insuring that the network buffers will not be overwritten before the contents can be captured. The capture circular buffer is typically quite large; if necessary, the capture enable bit can be turned off to assure that captured frames may be examined at leisure until the cows come home.

### ***Duplicate frames***

By monitoring the *identification* field—designed for identifying IP datagram fragments that belong to the same IP datagram in order to allow IP fragment reassembly to work—in the IP header of received frames, FMON can notice when duplicate frames occur by maintaining a small table of such data for frames recently received from the target node. This logic was installed at a time when we were experiencing such problems that turned out to stem from overloaded network hardware. There is no guarantee that a particular IP implementation would be compatible with this specific duplicate frame logic, but those that were tested implemented the standard by using an incrementing counter value for every IP datagram transmitted, a scheme that insures there will be no repeat until 65536 datagrams have been transmitted. It's simple and it works.

### ***Method of operation***

To perform an experiment in frame capture diagnostics, prepare suitable parameter values that will be needed, using Page E. The parameter that will likely require the most attention is the target node#. For IP frames, this will be a pseudo node# word that represents a UDP socket, as described earlier. A trial run may be needed to determine this value; use the NETFRAME diagnostics via Page F to get a handle on this. It can capture the occurrences of the last 237 frames received or transmitted. One can usually identify the frames of interest by the frame size and period of activity, so that one normally doesn't have to be quick enough to capture the actual frame contents using that program. Enter the node# thus determined as the target node# parameter value. It will be good for an hour or so, at least, because that is the timeout for ARP table entries. When that much time passes without any IP receive or transmit activity for that node, the ARP table entry will be cleared. The next time access occurs with the same node, it may quite possibly be assigned a different table entry than before, and will therefore have a different pseudo node#.

Choose the word offset used for frame capture according to what part of the frame contents needs to be captured. Select whether receive or transmit frames should be captured. Enable the capture flag bit. If it is already on, toggle it off and back on. This is optional, but it will cause the capture to begin at the start of the memory used for the purpose. The memory stores frames in a circular buffer mode; therefore, the last "MAX INDX" frames will be captured and available anyway, but it may be more convenient to know that the capturing starts at the beginning. When the capture enable bit transitions from 0 to 1, the internal logic resets the capture circular buffer pointer to the beginning.

Perform the experiment during which frames must be captured. If it is necessary to also capture frames relating to a second target node#, then install a second instance of the FMON local application. Obviously, different parameters must be used for the enable bit, although the capture enable Bit# parameter could be the same for both instances. But be careful not to use the same area of memory for the capture circular buffers needed by each instance!

When the condition is reached for which the frame data should be studied, one can turn off the capture enable Bit#. This will freeze the capturing. In some cases, this won't be necessary, as the condition that is reached may terminate the communication anyway. As an example, this tool was used to capture frame data so as to learn what caused snapshot plot activity to terminate. The condition reached was cancellation of the snapshot request and therefore termination of communication activity; no further frames would be captured anyway. It depends upon the exact nature of the experiment being performed.

To examine the frames captured, one can observe some of the static variables in use for this local application. Using the ptr shown on Page E, one can see the following picture on the Memory Dump Page for the time in which this was captured as an example:

```
      8 MEMORY DUMP     06/04/97 1055
      0509:044D9A 0904 0000 0000 0007
      0509:044DA2 0509 0091 0000 BF0B
      0509:044DAA 01CE 2000 0000 01CD
      0509:044DB2 0030 0000 0030 7300  Ptr into capture circular buffer
      0509:044DBA 0001 0001 0001 0001
      0509:044DC2 0101 0000 0000 0000
      0509:044DCA 0000 0000 0000 0000
      0509:044DD2 0000 0000 0000 0000

      0509:044DDA 000E 000E 0000 0000
      0509:044DE2 0000 0000 0000 0000
      0509:044DEA 4F06 BE76 2FE8 A059
      0509:044DF2 12CB 823A F126 DC96
      0509:044DFA AD8E C98E E08E F88E
      0509:044E02 1E8F 3B8F 558F 6E8F
      0509:044E0A 948F AC8F C48F DD8F
      0509:044E12 0390 568E 6E8E 878E
```

On the first "page," fourth line, in this example, the address 00307300 is the pointer to a 64-byte structure containing the date and up to 56 bytes of the contents of the last frame captured. Here is that structure:

```
      0509:307300 9706 0410 5506 0083  Time 6/4/97 10:55:06 cycle 0, 33ms
      0509:307308 008A 008A 00E6 DADE  UDP header (srcPort=dstPort=138)
      0509:307310 1102 12BD 83E1 7BD9  UDP datagram contents used for
      0509:307318 008A 00D0 0000 2045     NETBIOS datagram protocol
      0509:307320 4246 4145 4945 4A45
      0509:307328 4543 4143 4143 4143
      0509:307330 4143 4143 4143 4143
      0509:307338 4143 4143 4141 4100
```

Since the word offset parameter was $11 for this example, the captured frame data begins at the UDP header, assuming IP communications are used. In this example, the UDP port# used is 138. According to the RFC document that describes Internet Protocol Assigned Numbers, this port# is used for NETBIOS Datagram Service. The UDP length is 230 bytes. It was nothing of local interest, but in this example, pseudo node# E071 just happened to correspond to aphid.fnal.gov, presumably a PC. The last experiment using FMON in node0509 was done many days ago, after which the meaning of the pseudo node# changed to another node. Still, it serves here as an example. One can find the significance of the pseudo node# by looking up the entry in the IPARP table, currently based at $40E000 in all IRMs. At $40E070 we find the following ARP table entry:

```
      0509:40E070 00A0 2491 A138 0000
      0509:40E078 83E1 7BD9 0004 3A98
```

The port# block at $43A98 is as follows:

```
      0509:043A98 0060 0426 0000 0015  Port# memory block is type $15.
      0509:043AA0 002C 0003 0EBE 03CE
      0509:043AA8 0000 0000 0000 0000
      0509:043AB0 0000 0000 0000 0000
      0509:043AB8 0000 0000 008A 0000  Each entry in array is port#, use count
      0509:043AC0 0089 0000 0000 0000
      0509:043AC8 0000 0000 0000 0000
      0509:043AD0 0000 0000 0000 0000
      0509:043AD8 0000 0000 0000 0000
      0509:043AE0 0000 0000 0000 0000
      0509:043AE8 0000 0000 0000 0000
      0509:043AF0 0000 0000 0000 0000
```

In the port array, the port index 1 (lo 4 bits of $E071) corresponds to port 138. The IP address is 83E17BD9, or 131.225.123.217, which is aphid.fnal.gov. Looking at previous captured frames, it appears that this frame, presumably broadcast from aphid.fnal.gov, is received every 15 minutes. If this level of activity continues, this ARP table entry will not time out.

Duplicate frame diagnostics are shown on the second "page" of the first memory display. The two values of 000E followed by 0000 indicate that the recent time-stamp values and corresponding *identification* field values from the IP header. (The 000E values are indexes into the arrays of time-stamp and *identification* field values.) The fourth word of 0000 indicates that no duplicate frames received from the target node were tallied during the time since the capture enable bit was last set to 1. The time-stamp array is a byte array starting on the third line; the *identification* field value array begins on the fifth line.